**Universidad ORT Uruguay**

**Facultad de Ingeniería**

TePagoYa

Arquitectura de Software - Obligatorio

Entregado como requisito para la obtención del crédito Arquitectura de Software

Mariana Gonzalez - 156143

Mauricio Zito - 111725

Grupo N7A

Docentes: Andrés Calviño, Mathias Fonseca

2018

# Índice

[Índice 2](#_Toc530857737)

[1. Introducción 4](#_Toc530857738)

[2. Propósito 4](#_Toc530857739)

[3. Antecedentes 4](#_Toc530857740)

[3.1. Propósito del sistema 4](#_Toc530857741)

[3.2. Objetivos de Arquitectura 4](#_Toc530857742)

[3.2.1. Requerimientos Funcionales 4](#_Toc530857743)

[3.2.2. Resumen de requerimientos no funcionales 8](#_Toc530857744)

[3.2.3. Restricciones 9](#_Toc530857745)

[4. Documentación de Arquitectura 10](#_Toc530857746)

[Vista de Módulos del sistema: 10](#_Toc530857747)

[Presentación primaria: 10](#_Toc530857748)

[Catálogo de elementos: 11](#_Toc530857749)

[Justificación de diseño 12](#_Toc530857750)

[Vista de usos (con librerías de clase) 13](#_Toc530857751)

[Representación primaria 13](#_Toc530857752)

[Catálogo de elementos 13](#_Toc530857753)

[Comportamiento 13](#_Toc530857754)

[Justificación de diseño 13](#_Toc530857755)

[Vita de usos (TePagoYa y aplicaciones) 13](#_Toc530857756)

[Representación primaria 13](#_Toc530857757)

[Catálogo de elementos 13](#_Toc530857758)

[Justificación de diseño 13](#_Toc530857759)

[Vista de componentes y conectores 13](#_Toc530857760)

[Vista de Arquitectura de alto nivel 13](#_Toc530857761)

[Representación primaria 13](#_Toc530857762)

[Catálogo de elementos 13](#_Toc530857763)

[Justificación de diseño 13](#_Toc530857764)

[Vista de arquitectura de Comercio 13](#_Toc530857765)

[5. Decisiones de diseño 14](#_Toc530857766)

[6. Aseguramiento de Calidad 14](#_Toc530857767)

[7. Anexo 15](#_Toc530857768)

[7.1. End Points 15](#_Toc530857769)

[7.2. Manual de Configuración 16](#_Toc530857770)

[7.2.1. Creación de Base de datos 16](#_Toc530857771)

[7.2.2. Scripts en la Base de Datos 17](#_Toc530857772)

# Introducción

El siguiente documento tiene como propósito describir la arquitectura planificada para el sistema desarrollado para TePagoYa. Se identificarán requerimientos funcionales y no funcionales del sistema que servirán como conductores de la arquitectura y se especificará la arquitectura del sistema implementado siguiendo el modelo Views&Beyond.

# Propósito

En el presente documento se da una visión global y comprensible de la arquitectura del sistema de TePagoYa. En él se va encontrar detalles de implementación, las principales decisiones de diseño tomadas y sus correspondientes justificaciones.

# Antecedentes

## Propósito del sistema

El sistema de TePagoYa tiene como propósito simplificar lo más posible la integración de los distintos sistemas de forma de facilitar los pagos.Se trata de una herramienta que provee

la funcionalidad de actuar de intermediario entre los cuatro diferentes actores (comercio, gateway, red y emisor), resolviendo la mayor parte de los desafíos de integración como pueden ser los diferentes formatos de fechas y números, campos obligatorios y opcionales, formatos de representación de datos, etc.

## Objetivos de Arquitectura

### Requerimientos Funcionales

* **RF1 – Compra en Comercio**

Un consumidor puede realizar una transacción de compra en un comercio, informando los siguientes datos:

● Tarjeta

○ Número

○ Vencimiento

○ Nombre de titular

○ Código de seguridad

● Dirección de Envío

○ Calle

○ Número de puerta

○ Ciudad

○ País

○ Código postal

● Dirección de Facturación (mismos datos que la de Envío)

● Monto de transacción

● Fecha de transacción

● Producto a comprar

○ Cantidad

○ Nombre

○ Categoría

En función de la categoría del producto a comprar, el comercio deberá seleccionar el gateway que procesa la compra y enviarle la solicitud. Por ejemplo, los electrodomésticos pueden ser procesados por el Gateway A y los alimentos por el Gateway B. Esto debe ser configurable sin necesidad de modificar el código de los comercios.

La respuesta hacia el consumidor debe informar si la transacción fue exitosa o no y un código de transacción en caso de necesitar devolverla eventualmente.

* **RF2 – Compra en Gateway**

Cuando un comercio envía la información pertinente para que el gateway procese un pago, este debe decidir a qué red enviar el pago según a cuál de ellas pertenece la tarjeta. La identificación de la red debe realizarse en tiempos menores a 10 ms en promedio bajo cargas de 1000 solicitudes por minuto.

* **RF3 - Compra en red**

Cuando un gateway envía una transacción a la red , esta realiza los controles de prevención de fraude y en caso de que pase, la envía al emisor para el paso final de aprobación. Para el análisis de fraude, se revisa la cantidad de transacciones que hizo esa tarjeta en las últimas 72 horas y si supera un límite determinado, la transacción se rechaza. Este límite debe poder ser alterado en tiempo de ejecución sin necesidad de interrumpir la operación de la red.

La identificación del emisor de la tarjeta debe cumplir con las mismas condiciones que para la identificación de la red.

* **RF4 – Compra en Emisor**  
  Cuando la red envía una transacción a ser aprobada en el emisor, éste debe confirmar que la tarjeta sea válida y el saldo de la cuenta del titular sea suficiente para el monto a aprobar. Para que una tarjeta sea válida, debe haber sido emitida por el emisor, pasar el algoritmo de Luhn, no estar vencida, ni bloqueada (por falta de pago), ni denunciada (por robo o pérdida).
* **RF5 – Devolución**  
  Deberá ser posible realizar la devolución de cualquier compra realizada previamente que no haya superado el límite de días definido en el emisor . Realizando todos los controles pertinentes, notificando y asegurando la consistencia de todas las aplicaciones involucradas.
* **RF6 – Chargeback**

El usuario puede realizar un desconocimiento de compra en su emisor . El emisor deberá informar al comercio de este evento para que ejecute las acciones administrativas pertinentes. Se debe tener en cuenta que el comercio puede llegar a demorar un tiempo significativo en encontrar la transacción original ya que se puede hacer un chargeback de hasta 6 meses atrás.

* **RF7 – Cierre de lotes**

El comercio le solicita diariamente al gateway la transacción de cierre de lotes, en donde se informa al máximo detalle los movimientos de dinero que hubo en su cuenta en ese día (cerrando a determinada hora acordada). Esta transacción debe responder en un tiempo promedio menor a 50 ms bajo cargas de 5000 rpm.

* **RF8 – Cumplimiento de PCI**

Toda aplicación debe cumplir con los estándares de seguridad de PCI al máximo nivel posible, pudiendo demostrar y justificar a cada momento el nivel de cumplimiento.

Ver [Anexo](#_Análisis_para_cumplimiento)

* **RF9 – Mecanismo de integración de aplicaciones (TePagoYa)**

Desarrollar la aplicación TePagoYa para resolver la integración e interoperabilidad de las aplicaciones

actuales y futuras. Toda aplicación que se registra en TePagoYa puede actuar como prestador de servicios y/o consumidor de servicios.

Para cada prestador de servicios se debe poder:

● Registrar una “interfaz de servicio” que permita acceder a su funcionalidad.

● Especificar propiedades sobre la forma de usar cada interfaz (por ejemplo, el formato de las estructuras de datos que se intercambian por la interfaz, o el tipo de comunicación).

Para cualquier consumidor de servicios de debe poder:

● Localizar una interfaz de un servicio registrado

● Facilitar la invocación de las funcionalidades del servicio a consumir.

* **RF-10: Gestión de errores y fallas**

El sistema debe proveer suficiente información, de alguna forma, que permita conocer el detalle de las tareas que se realizan. En particular, en el caso de ocurrir una falla o cualquier tipo de error, es imprescindible que el sistema provea toda la información necesaria que permita a los administradores

hacer un diagnóstico rápido y preciso sobre las causas. Se espera que la solución contemple la posibilidad de poder cambiar las herramientas o paquetes concretos que se utilicen para producir esta información, así como reutilizar esta solución en otras aplicaciones, con el menor impacto posible en el código.

* **RF-11: Autenticación de operaciones**

Actualmente, cualquier usuario con acceso a la red informática donde están conectados los servidores de aplicaciones puede invocar libremente cualquier operación expuesta por las API de backend de cualquier aplicación (imagine un usuario ejecutando llamadas http usando Postman o cualquier otra herramienta).

Para toda aplicación “prestadora de servicios” que se registre en TePagoYa , se requiere que todas sus operaciones publicadas deben ser invocadas únicamente a través de TePagoYa . Es decir que toda invocación a una operación de un servicio registrado que se realice por fuera de TePagoYa se debe considerar como no legítima.

De igual forma, toda invocación a TePagoYa debe provenir de una aplicación “consumidora de servicios” legítima. Por ejemplo, realizar una invocación http por Postman a TePagoYa para consumir un servicio debe considerarse como un acceso no legítimo.

### Resumen de requerimientos no funcionales

|  |  |  |
| --- | --- | --- |
| **Id Requerimiento** | **Atributo de Calidad** | **Descripción** |
| **RQA1** | **Modificabilidad** | En función de la categoría del producto a comprar, el comercio deberá seleccionar el Gateway que procesará la compra y enviarle la solicitud. Esto debe ser configurable **sin necesidad de modificar el código de los comercios** |
| **RQA2** | **Eficiencia** | La identificación de la red debe realizarse en **tiempos menores a 10 ms en promedio bajo cargas de 1000 solicitudes por minuto.** |
| **RQA3** | **Modificabilidad** | Para el análisis de fraude, se revisa la cantidad de transacciones que hizo esa tarjeta en las últimas 72 hrs y si supera un límite determinado, la transacción se rechaza. **Este límite debe poder ser alterado en tiempo de ejecución sin necesidad de interrumpir la operación de la red.** |
| **RQA4** | **Seguridad** | Deberá ser posible realizar la devolución de cualquier compra realizada previamente que no haya superado el límite de días definido en el emisor. Realizando todos los controles pertinentes, notificando y **asegurando la consistencia de todas las aplicaciones involucradas.** |
| **RQA5** | **Eficiencia** | “El **comercio le solicita diariamente al gateway la transacción** de cierre de lotes…  **Esta transacción debe responder en un tiempo promedio menor a 50 ms bajo cargas de 5000 rpm**. |
| **RQA6** | **Modificabilidad/**  **Interoperabilidad** | Desarrollar la aplicación TePagoYa para **resolver la integración e interoperabilidad de las aplicaciones actuales y futuras**. Toda aplicación que se registra en TePagoYa puede actuar como prestador de servicios  y/o consumidor de servicios. |
| **RQA7** | **Seguridad** | “…suficiente información, de alguna forma, que permita conocer el detalle de las tareas que se realizan. En particular, en el caso de ocurrir una falla o cualquier tipo de error, es imprescindible que el sistema provea toda la información necesaria que **permita a los administradores**  **hacer un diagnóstico rápido y preciso sobre las causas .** |
| **RQA8** | **Modificabilidad** | Gestión de errores y fallas. “….espera que la solución contemple la posibilidad de poder **cambiar las herramientas o paquetes concretos que se utilicen para producir esta información, así como reutilizar esta solución en otras aplicaciones, con el menor impacto posible en el código.** |
| **RQA9** | **Seguridad** | Para toda aplicación “prestadora de servicios” que se registre en TePagoYa , se requiere que todas sus operaciones publicadas deben ser invocadas únicamente a través de TePagoYa . Es decir **que toda invocación a una operación de un servicio registrado que se realice por fuera de TePagoYa se debe considerar como no legítima**.  De igual forma, **toda invocación a TePagoYa debe provenir de una aplicación “consumidora de servicios” legítima.** |
| **RQA10** | **Seguridad** | Cumplimiento PCI- Proteger los datos almacenados de los titulares de las tarjetas |
| **RQA11** | **Seguridad** | Cumplimiento PCI- [Cifrar](https://searchsecurity.techtarget.com/definition/encryption) la transmisión de los datos de los titulares de tarjetas a través de redes públicas abiertas. |
| **RQA12** | **Seguridad** | Cumplimiento PCI- Limitar el acceso a los datos de los titulares, únicamente a lo que los negocios necesiten saber. |

### Restricciones

* La implementación del backend debe desarrollarse en NodeJS utilizando las tecnologías vistas en el curso. Es opcional y abierta la elección de packages que puedan ayudar al desarrollo, teniendo que justificar la elección en la documentación.
* Todo el código fuente, documentación, archivos de configuración o cualquier otro artefacto referido al desarrollo de los prototipos debe gestionarse en el repositorio Git asignado.
* Las API expuestas por los comercios deben ser REST. Para las comunicaciones entre otros servicios tiene libertad de seleccionar el mecanismo.
* Fecha de entrega: 29-11-2018

# 

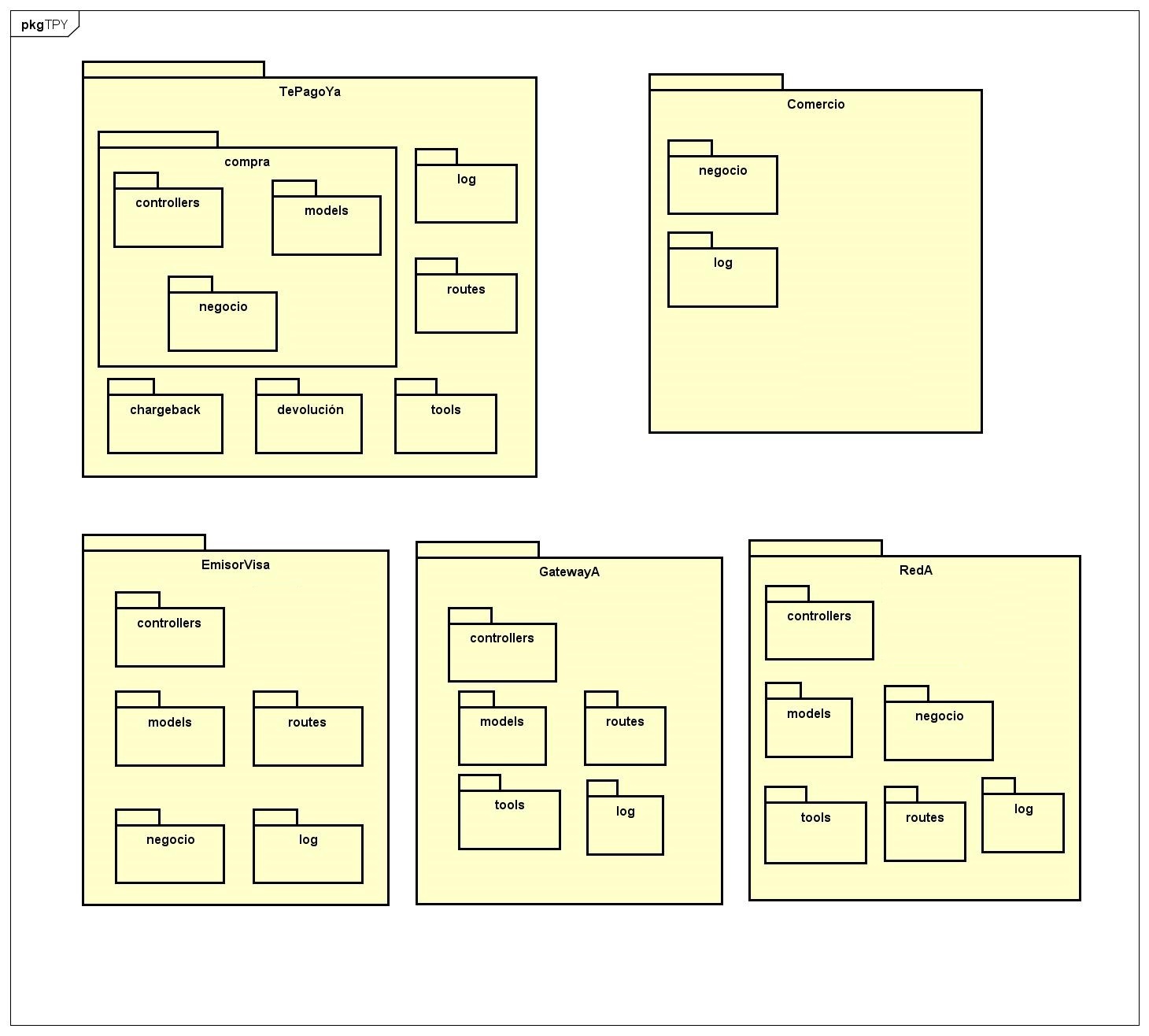
# 

# Documentación de Arquitectura

A continuación, se describe la arquitectura del sistema de TePagoYa desde distintos puntos de vista a través de componentes y conectores, módulos y asignación.

## Vista de Módulos del sistema:

### Presentación primaria:



### Catálogo de elementos:

|  |  |
| --- | --- |
| **Elemento** | **Responsabilidad** |
| TePagoYa.compra.controllers | Encargado de recibir las llamadas desde routers (de rutas conocidas) e indicar los procesos que se ejecutarán, en algunos casos resolviendo él mismo y en otros apoyándose en negocio para solucionar el problema o devolver información requerida. |
| TePagoYa.compra.models | Posee la definición de los esquemas de Mongoose utilizados en la implementación, se encarga de mantener la persistencia con la base de datos, existe un archivo js por cada uno de los esquemas. |
| TePagoYa.compra.negocio | Es donde se concentran las operaciones de resolución propios del negocio, como por ejemplo realizar los ajustes que requiere cada cliente inscripto, resolver el próximo destino (por el nombre del servicio registrado en la base de datos) y verificar en la base de datos si la referencia de la tarjeta de crédito (primeros cuatro caracteres) es válido para los datos recibidos. |
| TePagoYa.log | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución y tiene desde la funcionalidad para loguear eventos del sistema como errores (comunes y fatales) en éste último caso valiéndose de las herramientas ofrecidas por Log4Js mencionado anteriormente. Los archivos que se generan de forma automática se llaman TePagoYa\_Registro.log y TePagoYa\_Errores.log respectivamente. |
| TePagoYa.routes | En éste módulo se concentran los ruteos de servicios, lo que la aplicación está esperando y que debe hacer con cada uno de ellos. Cada ruta resuelta, está mapeada con un procedimiento expuesto y ofrecido por controllers. |
| TePagoYa.tools | En éste módulo transversal a la solución, se encuentran herramientas que centralizan actividades comunes como por ejemplo un log en pantalla de consola, procesos de conversión de información y criptografiar mediante la librería “crypto” provista por Node.js. |
| TePagoYa.devolucion | El proceso de devolución (no implementado) estaría concentrado en éste módulo, junto con los elementos de soporte necesarios (controllers, negocio y modelos) |
| TePagoYa.chargeback | El proceso de devolución (no implementado) estaría concentrado en éste módulo, junto con los elementos de soporte necesarios (controllers, negocio y modelos) |
| Comercio.negocio | Contiene una única responsabilidad, en función de la categoría del producto vendido, resuelve el Gateway donde derivará la llamada. Que por el **RQA1** se optó por utilizar un archivo de configuración categoriasXGateway.ini. |
| Comercio.log | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución y tiene desde la funcionalidad para loguear eventos del sistema como errores (comunes y fatales) en éste último caso valiéndose de las herramientas ofrecidas por Log4Js mencionado anteriormente. Los archivos que se generan de forma automática se llaman Comercio\_Registro.log y Comercio\_Errores.log respectivamente. |
| GatewayA.controllers | Encargado de recibir las llamadas desde routers (de rutas conocidas) e indicar los procesos que se ejecutarán, en algunos casos resolviendo él mismo y en otros apoyándose en otros módulos para solucionar el problema o devolver información requerida. |
| GatewayA.models | Posee la definición de los esquemas de Mongoose utilizados en la implementación, se encarga de mantener la persistencia con la base de datos, existe un archivo js por cada uno de los esquemas. En éste caso:  Redxcodigo: donde se relacionan los primeros 4 caracteres de los tipos de tarjeta de crédito aceptada y la red a donde debe ir en el siguiente paso) |
| GatewayA.tools | Se encuentran herramientas que son transversales a la solución, sirviendo de apoyo a tareas de conversión de datos y controles. |
| GatewayA.routes | En éste módulo se concentran los ruteos, lo que la aplicación está esperando y lo que debe hacer con cada uno de ellos, mapeando la llamada con el controlador correspondiente y su función. |
| GatewayA.log | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución y tiene desde la funcionalidad para loguear eventos del sistema como errores. |
| RedA.controllers | Encargado de recibir las llamadas desde routers (de rutas conocidas) e indicar los procesos que se ejecutarán, en algunos casos resolviendo él mismo y en otros apoyándose en negocio para solucionar el problema o devolver información requerida. |
| RedA.models | Posee la definición de los esquemas de Mongoose utilizados en la implementación, se encarga de mantener la persistencia con la base de datos, existe un archivo js por cada uno de los esquemas. En éste caso:  EmisorXCodigo: Donde resuelve el emisor que corresponde por el tipo de tarjeta.  TransxTarjeta: donde se registra las transacciones por cada tarjeta, cumpliendo con las normas de PCI, se registra exclusivamente el hash de cada tarjeta para utilizar en la verificación de intentos de fraude descritos en el REQ-3. |
| RedA.tools | Se encuentran herramientas que son transversales a la solución, sirviendo de apoyo a tareas de conversión de datos y controles. |
| RedA.routes | En éste módulo se concentran los ruteos, lo que la aplicación está esperando y lo que debe hacer con cada uno de ellos, mapeando la llamada con el controlador correspondiente y su función. |
| RedA.log | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución. |
| RedA.models | Encargado de recibir las llamadas desde ruters (de rutas conocidas) e indicar los procesos que se ejecutarán, en algunos casos resolviendo él mismo y en otros apoyándose en negocio para solucionar el problema o devolver información requerida. |
| EmisorVisa.controllers | Posee la definición de los esquemas de Mongoose utilizados en la implementación, se encarga de mantener la persistencia con la base de datos, existe un archivo js por cada uno de los esquemas. En éste caso:  Tarjetas: Registro de tarjetas de crédito autorizadas, código de seguridad y su fecha de vencimiento. |
| EmisorVisa.models | Se encuentran herramientas que son transversales a la solución, sirviendo de apoyo a tareas de conversión de datos y controles. |
| EmisorVisa.tools | En éste módulo se concentran los ruteos, lo que la aplicación está esperando y lo que debe hacer con cada uno de ellos, mapeando la llamada con el controlador correspondiente y su función. |
| EmisorVisa.routes | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución y tiene desde la funcionalidad para loguear eventos del sistema como errores (comunes y fatales) en éste último caso valiéndose de las herramientas ofrecidas por Log4Js mencionado anteriormente. Los archivos que se generan de forma automática se llaman Emisor\_Registro.log y Emisor\_Errores.log respectivamente. |
| EmisorVisa.log | En éste módulo de concentra todo lo relacionado al logueo, es transversal a toda la solución y tiene desde la funcionalidad para loguear eventos del sistema como errores (comunes y fatales) en éste último caso valiéndose de las herramientas ofrecidas por Log4Js mencionado anteriormente. Los archivos que se generan de forma automática se llaman Comercio\_Registro.log y Comercio\_Errores.log respectivamente. |

### Justificación de diseño

Nuestro proyecto está dividido en paquetes, se eligió arquitectura basada en módulos para favorecer el reúso común, apuntando al principio de cohesión de paquetes (PCP), por lo tanto, decidimos que aquella clase que usa otra clase, deben estar juntas en el mismo paquete.

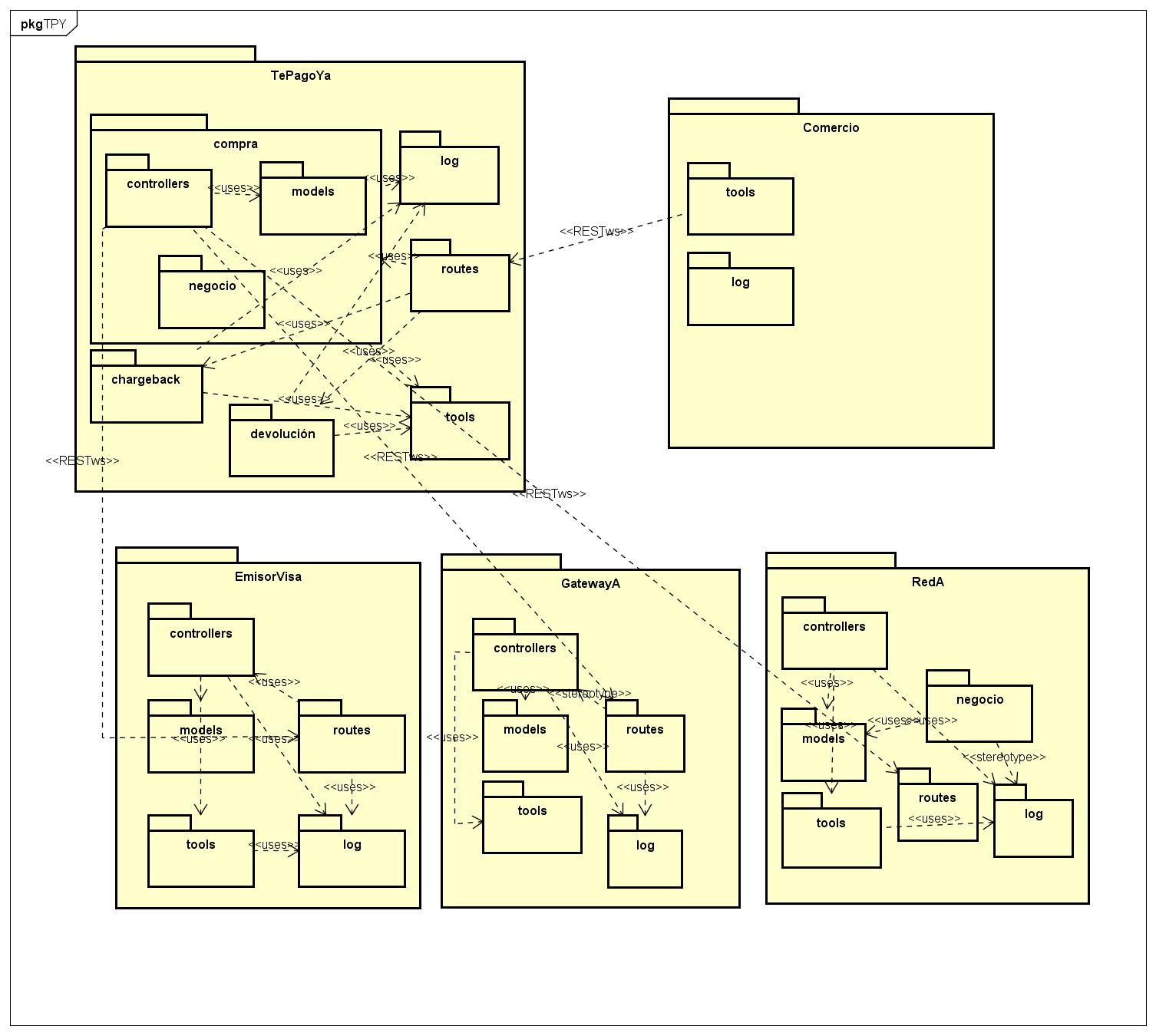
La forma en que lo armamos fue la siguiente: Armamos una estructura lógica de paquetes, agrupando por las funcionalidades que pensábamos agrupar, en primera instancia las que ya están definidas por la propia forma de trabajar del mercado y la planteada en el propio proyecto, luego fue necesario ubicar las clases que tienen función similar dentro de cada paquete.

La forma en que se diseñó cada paquete fue tomando en cuenta los síntomas de degradación más comunes e intentar minimizarlos:

|  |  |
| --- | --- |
| Rigidez | Cada paquete puede ser modificado de forma independiente, siempre y cuando respete las funcionalidades para las que fue construido, así también como la información devuelta. |
| Fragilidad | Cada paquete posee máxima cohesión y dependencia, por lo que los cambios que ocurran dentro del propio paquete, solo afectarán al funcionamiento del mismo, eventualmente a los indicadores de calidad. |
| Inmovilidad | Cada paquete fue creado para cumplir con una tarea específica, respetando lo indicado en las características del mismo, ya que es un estándar, por lo que permite la reutilización sin inconvenientes. |

## Vista de usos (TePagoYa y Aplicaciones)

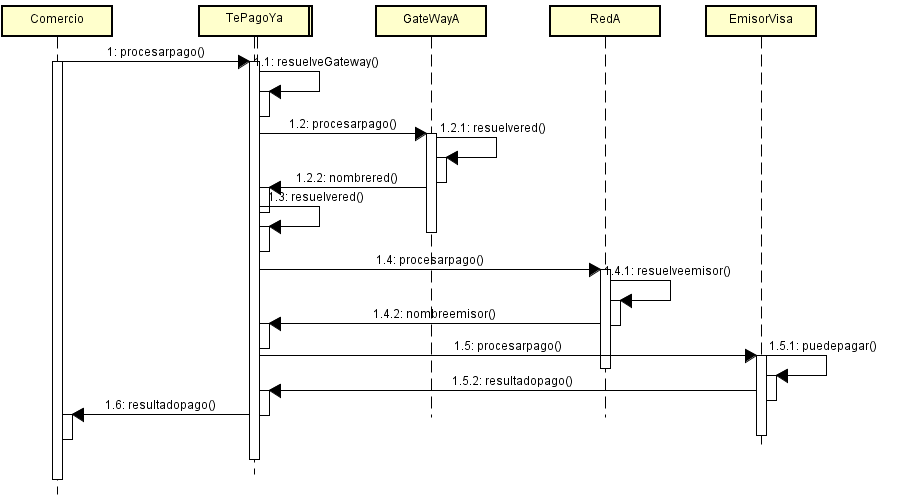
### Representación primaria



### Catálogo de elementos

Los elementos presentes en esta vista ya se especificaron en catálogos de elementos de las vistas anteriores.

### Comportamiento



### Justificación de diseño

De la vista anterior se puede ver los usos internos de los paquetes de las aplicaciones y las comunicaciones entre las mismas. En primer lugar, se observa que ninguna de las aplicaciones de TPY (Comercio,EmisorVisa,RedA y GatewayA) hablan entre sí directamente; se comunican únicamente con TePagoYa que es el módulo encargado de manejar las comunicaciones entre dichas aplicaciones. Esto apunta a cumplir con el RQA6 que además está estrechamente relacionado a uno de los requerimientos funcionales principales.

Se favorece la interoperabilidad de las aplicaciones y la modificabilidad del sistema al lograr desacoplar las aplicaciones limitándolas únicamente a comunicarse con TePagoYA Esto tiene sentido dado que cada aplicación no tiene que conocer las formas de comunicación con todas las demás y sus respectivas implementaciones para poder relacionarse. Por el contrario, conoce únicamente cómo hablar con TePagoYa y así se desacopla del resto.

Todas las aplicaciones se comunican con TePagoYa a través de web services REST ya sea para registrarse y publicar sus servicios o para realizar request. La decisión tiene que ver con que los integrantes del equipo ya conocían REST y que es eficiente y rápida. Más aún permite recibir los datos en diferentes formatos y admite respuestas de manera sencilla.

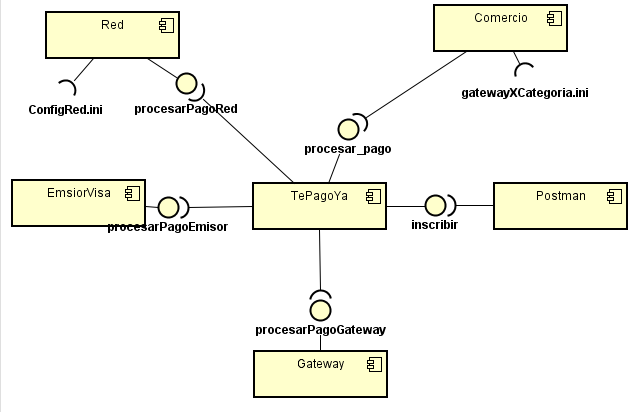
En referencia a las comunicaciones desde TePagoYa hacia las aplicaciones, se implementaron utilizando REST.

En referencia a los usos entre paquetes dentro de las aplicaciones, se cuidó de cumplir con los principios de diseño de dependencias acíclicas de forma de evitar ciclos entre las relaciones de uso de los paquetes. Dicha decisión tiene como propósito seguir favoreciendo la modificabilidad.

## Vista de componentes y conectores

### Vista de componentes y conectores de TePagoYa

#### Representación primaria



#### Catálogo de elementos

#### Justificación de diseño

El objetivo de ésta es observar la arquitectura a grandes rasgos y en un alto nivel. Los componentes que allí se mostrarían, en particular las cinco aplicaciones node.js, están compuestos a su vez por otros componentes.

El propósito de esta vista es mostrar la arquitectura de la aplicación, definida de acuerdo con el patrón SOA. Esta decisión está estrechamente relacionada con la funcionalidad esperada del sistema: una serie de aplicaciones interactuando entre sí de manera independiente a través de un módulo intermediario que las comunique y que permita la incorporación de nuevas aplicaciones con bajo impacto de cambio.

El patrón SOA describe una colección de componentes distribuidos que proveen y/o consumen servicios. Pretende ser una solución para aquellos sistemas donde se busque soportar la interoperabilidad de componentes que se encuentren en distintas plataformas y que están implementados en distintos lenguajes.

El patrón SOA incluye diferentes elementos posibles. En el caso de este sistema, los elementos que se tienen son los siguientes:

* TePagoYa es el componente que media entre el resto de las aplicaciones. El mismo es una versión simplificada y reducida de lo que sería un ESB en SOA. TePagoYa se encarga de recibir y rutear los mensajes entre las aplicaciones consumidoras y proveedoras, realiza verificaciones para comprobar que las llamadas a métodos son correctas. TePagoYa realiza chequeos de autenticación de las aplicaciones por lo que también favorece la seguridad. Todas estas características de TePagoYa tienen como propósito ayudar a cumplir los requerimientos no funcionales RQA4, RQA6 y RQA9.
* Comercio, EmisorVisa, RedA y GatewayA

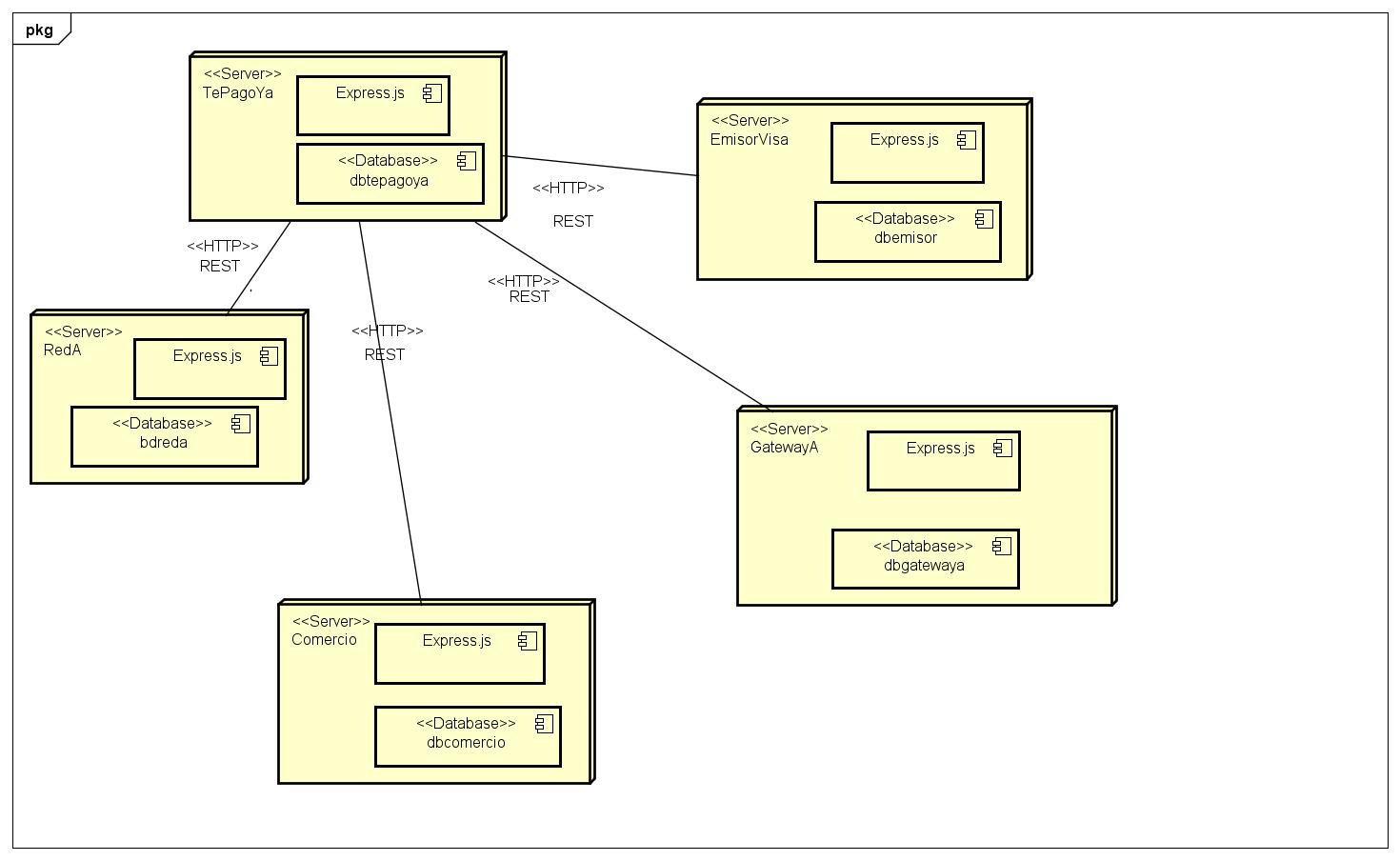
Por otro lado se puede observar que el componente RedA lee un archivo ConfiRed.Ini donde se parametriza la cantidad de transacciones límite antes de que la transacción no se considere fraudulenta. De esta forma se da cumplimiento al atributo de calidad RQA3.

De la misma forma Comercio lee el archivo gatewayXCategoria.Ini donde se parametriza a qué Gateway se corresponde cada categoría. De esta forma se da cumplimiento al atributo de calidad RQA1.

## Vista de despliegue

### Vista de asignación

#### Representación primaria



#### Catálogo de elementos

|  |  |
| --- | --- |
| **Catálogo de elementos** | **Responsabilidad** |
| TePagoYa | Es el ambiente de ejecución de TePagoYa |
| Express.js | Es el artefacto que contienen los resources de la aplicación con los web services correspondientes de TePagoYa. |
| dbtepagoya | Es donde persiste la información de los servicios relacionados con el proceso que controla TePagoYa. |
| GatewayA | Es el ambiente de ejecución de GatewayA |
| Express.js | Es el artefacto que contienen los resources de la aplicación con los web services correspondientes al GatewayA. |
| dbgatewaya | Es donde persiste la información de los servicios relacionados con el proceso que controla el GatewayA. |
| Comercio | Es el ambiente de ejecución de Comercio. |
| Express.js | Es el artefacto que contienen los resources de la aplicación con los web services correspondientes al Comercio. |
| dbcomercio | Es donde persiste la información de los servicios relacionados con el proceso que controla el Comercio. |
| EmisorVisa | Es el ambiente de ejecución de EmisorVisa. |
| Express.js | Es el artefacto que contienen los resources de la aplicación con los web services correspondientes al EmisorVisa. |
| dbemisorvisa | Es donde persiste la información de los servicios relacionados con el proceso que controla el EmisorVisa. |
| RedA | Es el ambiente de ejecución de RedA. |
| Express.js | Es el artefacto que contienen los resources de la aplicación con los web services correspondientes a la RedA |
| dbreda | Es donde persiste la información de los servicios relacionados con el proceso que controla la RedA |

#### Justificación de diseño

Un primer ítem para observar en la vista anterior es que se está desplegando cada una de las aplicaciones en un servidor distinto. Este despliegue en distintos servidores uno de los objetivos que busca es aumentar la performance para dar cumplimiento con los requisitos RQA5 y RQA2.

El hecho de que todas las aplicaciones estén desplegadas de manera independiente permite que, ante un cambio en alguna de ellas, no haya que volver a desplegar todas las aplicaciones del sistema, lo que nos aumenta la modificabilidad de la solución.

Se decidió colocar la base de datos del lado del servidor para mejorar la performance de la respuesta a efecto de dar cumplimiento de los requisitos antes mencionados.

Cabe mencionar que en una situación real dado el negocio los servidores necesariamente van a estar desplegados de forma independiente.

# Cumplimiento de requerimientos no funcionales

|  |  |  |  |
| --- | --- | --- | --- |
| **Id Requerimiento** | **Atributo de Calidad** | **Tácticas/**  **Patrones** | **Solución** |
| **RQA1** | **Modificabilidad** | Diferir Enlace / Interpretar parámetros | Mediante un archivo de configuración en el servidor del comercio se indicará cual es el Gateway resultante dependiendo de la categoría del producto. |
| **RQA2** | **Eficiencia** | Manejo de Recursos/ Mantener múltiples copias de datos | Se cachea la colección de referencia que vincula la red con el código de la tarjeta (cuatro primeros números) . Esta caché se implementó en el Gateway y se actualiza cada 500 transacciones. |
| **RQA3** | **Modificabilidad** | Diferir Enlace / Interpretar parámetros | Mediante un archivo de configuración en la servidor de la red se parametrizará la cantidad de transacciones límite para que la aplicación lo interprete como fraude. |
| **RQA4** | **Seguridad** | Detectar Ataques/ Verificar la integridad del mensaje | Todas las transacciones son identificadas mediante un check sum que se incorpora en la primer llamada y acompaña todo el proceso. |
| **RQA5** | **Eficiencia** | Manejo de Recursos/ Mantener múltiples copias de datos | Se cachean todas las transacciones diarias de todos los comercios. Esta caché se implementó en el Gateway y se actualiza cada diariamente. |
| **RQA6** | **Modificabilidad/**  **Interoperabilidad** | Usar intermediario Patrón SOA | Por la forma en que se implementó TePagoYa, funciona como intermediario en la comunicación entre las distintas aplicaciones. Las mismas pueden estar escritas en lenguajes diferentes o utilizar distintos protocolos de comunicación y TePagoYa hará que la interacción sea uniforme, convirtiendo los datos de un protocolo a otro según cuál utilice cada aplicación. De esta forma se logra favorecer la interoperabilidad del sistema y en consecuencia aumenta la modificabilidad. |
| **RQA7** | **Seguridad** | Mantener registro de auditoría | Se implementó un módulo de registro de logs para llevar registro de las acciones que se realizan y los errores que ocurren de forma de poder realizar un diagnóstico rápido y preciso siempre que sea necesario. Mantener un registro de auditoría es la táctica que se utilizó para favorecer la seguridad del sistema. |
| **RQA8** | **Modificabilidad** | Encapsular  Dividir módulos  Incrementar coherencia semántica | El módulo de logs se implementó como una biblioteca de clases y todo el comportamiento asociado a log4js (herramienta utilizada para los logs en este caso) quedo dentro de dicho módulo. Por ende, de querer cambiarse la herramienta para hacer los logs, basta con modificar la implementación de los métodos del módulo de Logging. El resto de las aplicaciones no se verían afectadas. |
| **RQA9** | **Seguridad** | Identificar actores  Autenticar actores  Autorizar actores | TePagoYa le asigna una contraseña a cada aplicación en el momento que se registra. La misma se utilizará en el envío de mensajes entre TePagoYa y las aplicaciones para verificar que las llamadas son legítimas. Por último, si la autenticación es satisfactoria se podrá dar autorización o no para las distintas acciones según corresponda. Cualquier invocación que no responda a esta contraseña se considerará ilegítima. |
| **RQA10** | **Seguridad** | Resistir Ataques/ Encriptar Información | En el servidor de la red que es donde se almacenan los datos de la tarjeta de crédito se almacenan encriptados. |
| **RQA11** | **Seguridad** | Resistir Ataques/ Encriptar Información | Para esto deberíamos encriptar y desencriptar la información en cada transmisión de estos datos. Por un tema de tiempos esto no quedó implementado. |
| **RQA12** | **Seguridad** | Resistir Ataques/ Separar Entidades | Para esto deberíamos separar la información sensible y enviar a través de TePagoYa lo que cada servidor necesita saber. Por temas de tiempos esto no se hizo. |

# Aseguramiento de Calidad

Para mejorar la mantenibilidad del sistema se usaron varias técnicas que facilitan la introducción de cambios y disminuyen los defectos del sistema.

Se utilizó las buenas prácticas de Clean Code para tener un código mantenible y reusable.

Por otro lado, se mejoró el código con el uso de CheckStyle.

Se siguieron las buenas prácticas de restful recomendadas en la siguiente URL.

https://[blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/](http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/)

# Anexo

## Análisis para cumplimiento de normas PCI

Se hace un breve resumen de los requisitos PCI DSS y su cumplimiento en nuestro trabajo:

1. Instalar y mantener una configuración de [firewall](https://searchsecurity.techtarget.com/definition/firewall) para proteger los [datos](https://searchdatamanagement.techtarget.com/definition/data) de los titulares de tarjetas.

***Se entiende no corresponde al alcance del presente trabajo ya que los datos del emisor están en un sistema aparte. En nuestro trabajo no almacenaremos datos de los titulares.***

1. No utilizar los [valores predeterminados](https://whatis.techtarget.com/definition/default) suministrados por el proveedor para las [contraseñas](https://searchsecurity.techtarget.com/definition/password) del sistema y otros parámetros de seguridad. ***Se entiende no corresponde al alcance del presente trabajo.***
2. Proteger[los datos almacenados de los titulares de tarjetas](https://searchstorage.techtarget.com/definition/data-at-rest).

***Se genera el atributo de calidad RQA10 de seguridad para contemplar este punto.***

1. [Cifrar](https://searchsecurity.techtarget.com/definition/encryption) la transmisión de los datos de los titulares de tarjetas a través de redes públicas abiertas.

***Se genera el atributo de calidad RQA11 de seguridad para contemplar este punto.***

1. Usar y actualizar con regularidad el software antivirus.

***Se entiende no corresponde al alcance del presente trabajo. Se presupone que cada servidor tiene la seguridad adecuada.***

1. Desarrollar y mantener sistemas y aplicaciones seguras.

***Se entiende no corresponde al alcance del presente trabajo. Se presupone que cada sistema y aplicación tienen los niveles de seguridad requeridos. Nuestra recomendación en este punto es seguir los lineamientos de*** [***OWASP***](https://www.owasp.org/index.php/Main_Page)

1. Limitar el acceso a los datos de los titulares, únicamente a lo que los negocios necesiten saber.

***Se genera el atributo de calidad RQA12 de seguridad para contemplar este punto.***

1. Asignar una identificación única a cada persona con acceso a una computadora.

***Se entiende no corresponde al alcance del presente trabajo.***

1. Restringir el acceso físico a los datos de los titulares de tarjetas.

***Se entiende no corresponde al alcance del presente trabajo.***

1. Rastrear y monitorear todo acceso a los recursos de la red y a los datos de titulares de tarjetas.

***En nuestra solución se generó un log de auditoría que cumple con este requerimiento RQA7.***

1. Probar con regularidad los sistemas y procesos de seguridad.

***Se entiende no corresponde al alcance del presente trabajo Nuestra recomendación en este punto es seguir los lineamientos de*** [***OWASP***](https://www.owasp.org/index.php/Main_Page)

1. Mantener una política que aborde la seguridad de la información.

***Se entiende no corresponde al alcance del presente trabajo Nuestra recomendación en este punto es seguir los lineamientos de*** [***OWASP***](https://www.owasp.org/index.php/Main_Page)

Para un análisis exhaustivo se sugiere la lectura del documento: https://es.pcisecuritystandards.org/\_onelink\_/pcisecurity/en2es/minisite/en/docs/PCI\_DSS\_v3.pdf